반도체디스플레이기술학회지 제22권 제3호(2023년 9월)
Journal of the Semiconductor & Display Technology, Vol. 22, No. 3. September 2023.

TP-Sim: 트레이스 기반의 프로세싱 인 메모리 시뮬레이터

김정근[*†]
[*†]경북대학교 컴퓨터학부

TP-Sim: A Trace-driven Processing-in-Memory Simulator

Jeonggeun Kim[*†]
[*†]School of Computer Science and Engineering, Kyungpook National University

ABSTRACT

This paper proposes a lightweight trace-driven Processing-In-Memory (PIM) simulator, TP-Sim. TP-Sim is a General Purpose PIM (GP-PIM) simulator that evaluates various PIM system performance-related metrics. Based on instruction and memory traces extracted from the Intel Pin tool, TP-Sim can replay trace files for multiple models of PIM architectures to compare its performance. To verify the availability of TP-Sim, we estimated three different system configurations on the STREAM benchmark. Compared to the traditional Host CPU-only systems with conventional memory hierarchy, simple GP-PIM architecture achieved better performance; even the Host CPU has the same number of in-order cores. For further study, we also extend TP-Sim as a part of a heterogeneous system simulator that contains CPU, GPGPU, and PIM as its primary and co-processors.

Key Words : Processing-in-Memory, Simulator, Trace-driven Simulator, Data-intensive processing, Memory Architecture

1. 서 론

심층 신경망 기반의 추론 및 학습, 인메모리 프로세싱, 그래프 프로세싱 등 데이터 집약적 연산을 수행하는 워크로드들의 확산으로 인하여, 기존의 멀티코어 기반 병렬 컴퓨팅 환경에 있어서 주된 병목현상을 발생하는 구간이 연산 영역으로부터 메모리 및 스토리지와 같은 데이터 저장장치로 이동하고 있다.

이러한 데이터 집약적 프로세싱을 수행하는 워크로드를 구동하는 시스템에서의 성능적 저하를 막기 위하여, 최근 Intel Optane [1]과 같은 스토리지 클래스 메모리(Storage Class Memory)의 도입, AMD社의 CPU 적층형 3D V-cache [2]의 도입과 같은 시도가 이루어지고 있다. 그러나, 해당 장치들의 도입을 통해서는 폰 노이만 아키텍처(Von Neumann Architecture)의 근원적인 한계점인 연산 장치 및 데이터 저장장치 간의 데이터 이동에서 발생하는 지연시간은 해결이 불가하다 [3].

그러나 최근 데이터 근접 처리 (Near-data Processing) 기반의 컴퓨팅 시스템이 제안됨에 따라, 데이터가 저장된 장소에 근접한 곳에 프로세싱 유닛을 배치하여 데이터 이동 과정에서 발생하는 지연시간을 근원적으로 최소화할 수 있는 기법들이 연구되고 있다. 대표적으로, 삼성의 AxDIMM [4], 하이닉스의 AiM [5], 마이크론의 Hybrid Memory Cube (HMC) [6], UPMEM [7] 등 DRAM에 연산장치 (Processing Elements; PE)를 적층하거나, 메인 메모리 장치에 경량화 된 연산장치를 내장하는 방식을 통해서 데이터 및 메모리 집약적인 연산에 적합한 형태의 장치를 제안하고 있다.

하지만, 이 중 UPMEM을 제외하고는 현재 상용적으로 구동이 가능한 장치가 존재하지 않는 상황이다. 그리고 UPMEM과 같은 PIM 장치들의 경우, PIM 내부 구조의 변경 및 데이터 배치 및 관리 정책 변화 등에 따른 성능적

†E-mail: jeonggeun.kim@knu.ac.kr

변화를 관찰할 수 있는 등의 유연성을 제공하는 것이 주 목적이 아니기에, PIM 아키텍처 연구를 위한 플랫폼으로 써는 부적합하다.

즉, 다양한 측면에서의 PIM 분야 연구 수행을 위해서는, PIM 구조 변화에 따른 워크로드 구동 상의 성능 및 에너지 소모 변화 추이 등에 대한 측정 및 평가가 가능해야 하며, 장치의 기능적 수행에 대한 검증도 가능해야 한다. 따라서 유연하게 PIM 내부 구조를 변경할 수 있는 소프트웨어 기반의 PIM 아키텍처 시뮬레이터의 중요성이 대두되고 있다.

하지만, 기존 PIM 시뮬레이터들 중에서 MultiPIM [8] 및 ZSim+Ramulator [9]의 경우 PIM을 위한 독립적인 시뮬레이터가 아닌, 기존 x86 아키텍처 시뮬레이터인 ZSim [10]을 바탕으로 구동되는 형태 이기에, 그 확장성에서 단점이 존재한다. 그리고 명령어 트레이스 기반이 아닌 Intel Pin tool [11]을 통한 Dynamic Binary Instrumentation (DBI) 기반 구동을 통해 모델링을 수행하기에 수행 성능적 측면에서 약점이 존재한다.

본 연구에서는 다양한 PIM 하드웨어 구조에 대해서 유연한 시뮬레이션을 가능하게 하는 경량화된 트레이스 기반 시뮬레이터인 TP-Sim을 제안하고자 한다. TP-Sim은 범용적 PIM 장치 (General Purpose PIM; GP-PIM) 모델링을 통해, 해당 구조의 실행 시간, PIM vault 별 memory latency distribution, memory bandwidth, PE cache의 MPKI 및 hit ratio 등 다양한 성능적 지표를 제공할 수 있는 기능을 제공하고자 한다.

본 논문에서는 제안한 시뮬레이터를 통하여 STREAM 벤치마크 [12]를 구동하여, 여러 가지 형태의 컴퓨팅 시스템에 대한 성능 평가를 수행하였으며, 이를 통해 범용적 PIM 구조 평가를 위한 시뮬레이터로써 TP-Sim의 적합성을 검증하였다.

2. TP-Sim: Trace-driven PIM Simulator

2.1 PIM 구조 및 시뮬레이터 개요

본 연구는 다양한 형태의 프로세싱 인 메모리 (Processing-in-Memory, PIM) 구조에 대한 성능적 검증을 수행하기 위하여, 명령어 및 메모리 트레이스 기반의 PIM 아키텍처 시뮬레이터인 TP-Sim을 설계 및 구현하였다.

본 연구에서 제안한 시뮬레이터가 모사하고자 하는 PIM 구조는 Fig. 1에서 제시되어 있다. 해당PIM 아키텍처는 PIM Device, PIM Stack, Vault와 같은 계층 구조로 이루어져 있다. 해당 구조의 경우 일반적인 범용 목적을 위한 PIM (General-Purpose PIM; GP-PIM)의 기본형이라고 할 수 있는 HMC [6]에 기반한 것으로써, 기존의 여러 연구에서 제안 한 GP-PIM 모델들 과의 원활한 비교 수행을 위해 HMC기반 GP-PIM모델들 [6][8][13]의 공통적인 구성 요소를 기본형으로 갖추도록 설계하였다.

Vault는 여러 층으로 적층 된 메모리 어레이에 대해서, 균일한 파티션들로 나누어진 영역 중 하나를 의미한다. 각 메모리 어레이의 계층 별 Vault에 속한 partition들은 같은 Processing Element (Vault에 적재된 데이터에 대한 연산을 수행하는 logic)과 메모리 컨트롤러를 공유한다.

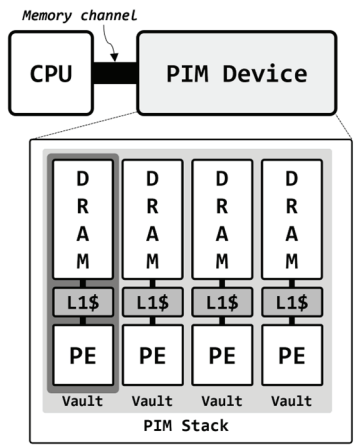

Fig. 1. Internal structure of Processing-in-Memory (PIM).

PIM Stack의 경우 일반적으로 PIM 칩 하나를 의미하며, 시스템 구성에 따라 여러 개의 PIM 칩들로 구성된 경우도 있다. 기존 연구 중 Multi-PIM [8] 및 3D-memory 기반의 NDP[13]와 같은 구조의 GP-PIM들의 경우, 해당 PIM 칩 간의 토폴로지를 고려하여, 시뮬레이션을 수행하는 것을 주된 기능으로 제공하였다.

본 논문에서 제안한 TP-Sim은 여러 개의 PIM 칩들로 구성된 PIM Device및 CPU + GPU + multiple PIM chips로 구성된 시스템에 대한 시뮬레이션이 아닌, GP-PIM 상에서의 CPU+PIM 간의 오프로딩 (offloading)의 효율성 평가 및 메모리 구조 및 정책 상의 성능적 평가를 위한, 경량화 트레이스 기반의 PIM시뮬레이터를 목표로 한다.

Fig. 1과 같이 TP-Sim은 PIM Device와 Host CPU를 동시에 구동할 수 있는 환경이며, Host CPU의 경우 비순차적 프로세서 (Out-of-Order Processor) 및 순차적 프로세서 (In-order Processor) 여부에 대한 설정과 Blocking/Non-blocking 캐시 메모리 구조 채택 여부에 대한 사용자 설정 기능을 제공한다.

2.2 TP-Sim 내부 구조 및 동작 흐름

본 연구는 2.1에서 제시한 PIM 구조를 모사할 수 있는 시뮬레이터인 TP-Sim의 설계 및 구현을 수행하였다. Fig. 2

와 같이 TP-Sim은 Intel Pin tool을 활용하여, 실제 Host 환경에서 수행한 워크로드의 명령어 및 메모리 트레이스를 추출하여 입력으로 사용한다. 각 Vault별로 수행하게 되는 메모리 관련 및 연산 수행 관련 명령어들에 대한 정보를 트레이스 파일들로 추출한고, 각 PIM Vault별 Processing Element (PE)가 해당 명령들을 명령어 큐에 적재한 후 처리를 수행한다.

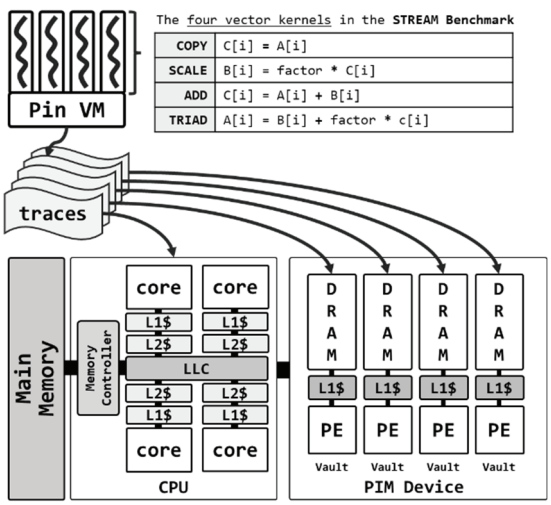

Fig. 2. TP-Sim's internal structure and the flow of simulation.

Host CPU의 경우 Fig. 2의 왼쪽 하단과 같이 L1, L2, L3 (Shared LLC)와 같은 complete cache 계층 구조를 가지며, 구성 설정에 따라 멀티 코어 시뮬레이션을 수행할 수 있다. PIM Device의 경우에는 Host CPU에서 수행하는 워크로드의 master thread로부터 생성된 thread들을 수행하는 방식으로 구동된다. 또한, 일반적인 GP-PIM 상에서 PIM Die size 및 전력 소모, 비용 문제 등을 종합적으로 고려하여 in-order 코어 및 blocking-cache를 채택하는 경향을 반영하여, 다음 장에서 수행한 성능 평가에서 TP-Sim의 경우 PIM Vault의 PE로써 경량화된 순차 프로세서를 사용한다. 해당 순차 프로세서의 경우 1-level 캐시 구조에 기반하고 있으며, 해당 캐시는 blocking-cache 구조를 채택하고자 경우 Miss Status Holding Register의 size를 1로 설정 한다.

3. 성능평가

3.1 실험환경 구축

제안한 시뮬레이터의 기능적 검증을 위하여 STREAM Benchmark [12]의 구동을 통해, 기존 CPU 및 DDR 메인 메모리 기반의 시스템과의 성능을 비교하였다.

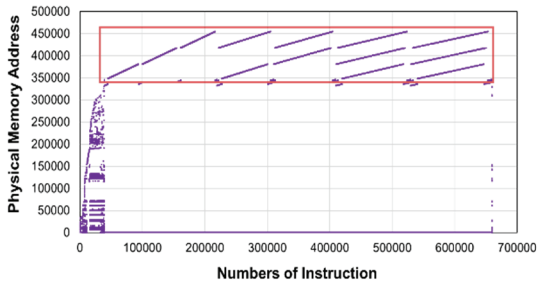

Fig. 3. Memory access patterns of STREAM Benchmark.

STREAM Benchmark의 경우 Fig. 2의 상단에서와 같이 4가지 주요 마이크로 커널로 구성되어 있다. 해당 커널들은 각각 복사, 상수 곱셈, 벡터간 덧셈, 상수 곱셈 후 벡터 합 연산과 같이, 기본적인 벡터 연산을 통해 높은 비율로 메모리 데이터를 요청하는 패턴을 보인다.

Fig. 3에서와 같이 STREAM Benchmark의 구동과정에서는 각 벡터(배열)에 대해서 순차적 패턴으로 접근하며, 하나 혹은 두 개의 배열에 대한 연산 결과를 다른 벡터에 저장하는 패턴을 보인다. 가장 처음 (1) A, B, C 벡터에 대한 초기화 패턴을 보이며, (2) 배열 A[i]의 내용을 C[i]에 복제하는 보이는 부분까지 Fig. 3에서 붉은색 사각형 내부에서 보이는 메모리 접근 패턴을 통해 시각적으로 확인이 가능하다. 이러한 STREAM 벤치마크의 경우 극단적으로 메모리 영역에 자주 접근하는 워크로드가 아키텍처에서 어떤 영향을 미치는 지에 대해서 확인할 때 유용한 성질을 지니고 있으며, DRAM 및 NVM 아키텍처에 대한 시뮬레이터의 기본적인 기능을 검증할 때 유용하게 활용이 가능하다.

따라서, 본 연구에서는 TP-Sim의 구동에 대한 검증을 수행하기 위하여, 상기 STREAM 벤치마크 대해 다음과 같은 시뮬레이션 결과들을 중점적으로 확인해보고자 한다. TP-Sim 상에서, (1) 기본적인 Execution time에 대한 성능적 비교 기능, (2) 각 Vault 별 구간별 평균 메모리 대역폭 산출 기능 (Bandwidth)을 점검하고자 한다.

더불어, 다음과 같이 총 세 가지 형태의 아키텍처에 대한 실험 진행을 통하여, PIM 아키텍처의 효용성에 대한 검증도 가능한지 확인하고자 한다.

(1) Host CPU + PIM 으로 구성된 이종시스템
(2) CPU 및 메모리 계층 구조 기반 시스템
(3) (1)에서의 Vault 개수 + Host CPU 코어 수 만큼의 멀티 코어로 구성된 (2)의 시스템

또한, 3.2 성능 평가에서는 다음 Table 1과 같은 시뮬레이션 설정을 기반으로 실험을 수행하였다.

Table 1. Simulation configuration for TP-Sim

	Component	Configuration
Host CPU	CPU (Performance)	Out-of-order core(s) 2.4GHz, 4-issue non-blocking cache
	CPU (Light-weight)	In-order core(s) 2.4GHz, 4-issue blocking cache
	Main Memory	DRAM Memory Scheduler: FR-FCFS t_{RP}=13, t_{CAS}=13, t_{RCD}=13, t_{WR} = 8, t_{RAS} = 30
PIM	PE	In-order core(s), 2.4 GHz blocking cache (MSHR size = 1) L1$ (hit latency: 3 cycles)
	Memory	4 vaults, 1.25 GHz t_{RP}=7, t_{CAS}=7, t_{RCD}=7, t_{WR} = 9, t_{RAS} = 14 (cycles)

Table 1에서는 기존 HMC [6] 스펙 및 타 GP-PIM에서 주로 구성된 PIM 구성 파라메터 [8][13][14]를 기반으로 성능 평가를 하기 위해 구성을 진행하였으며, 해당 파라메터의 경우 별도의TP-Sim configuration 파일 수정 등을 통해 유동적으로 변경 가능하도록 구성하였다.

3.2 성능 평가

다음은 3.1 실험환경 구축에서 제시한 환경에서의 STREAM Benchmark에 대한 구동 결과이다.

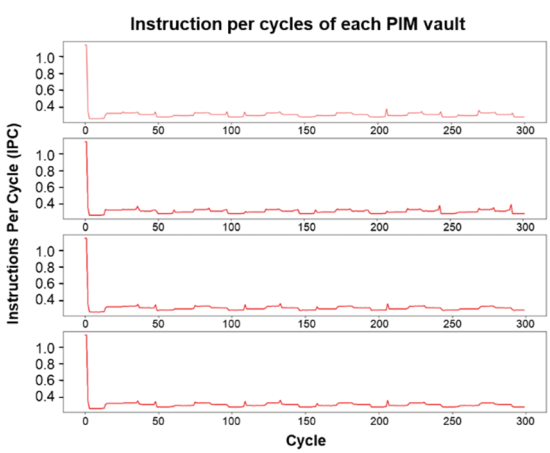

Fig. 4. Instructions per cycle of each PIM vault.

Fig. 4는 각 PIM vault 별 성능 지표 중 하나인 사이클 당 명령어 수행 횟수 (IPC)에 대한 결과이다. 여러 PIM vault에서 동시적인 STREAM 벤치마크 구동을 위해서, OpenMP로 구현된 STREAM 벤치마크 [12]를 PIM의 vault 수에 맞추어 병렬화 하여 실행한 후 명령어 및 메모리 트레이스를 추출하였으며, 이를 TP-Sim에서 구동하였다. 해당 그래프의 X축은 사이클 구간을 의미하며, Y축은 IPC 값을 의미한다.

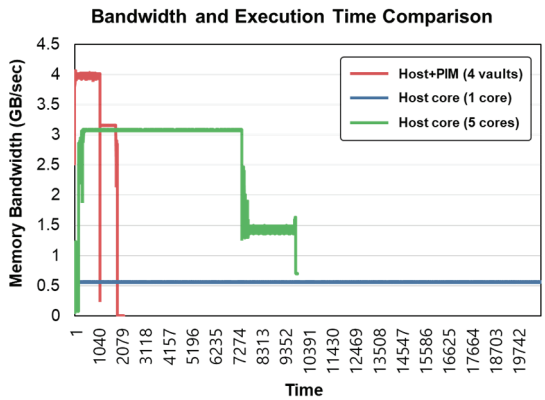

Fig. 5. Bandwidth and Execution time comparison between (1) Host + PIM, (2) Host core (single core), and (3) 5-core-based Host CPU.

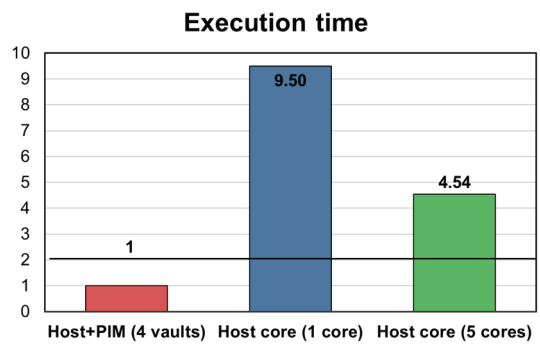

Fig. 6. Execution time of various PIM and CPU configurations.

Fig. 5의 경우에는 3.1 실험환경 구축에서의Host CPU와 (싱글 코어 및 멀티 코어), Host CPU + PIM으로 구성된 시스템 상에서 실험한 STREAM 벤치마크 구동 시간에서의 차이와 메모리 대역폭 (GB/s) 측정에 대한 결과이다. X축은 수행 시간을 (24000 clock 단위), Y축은 메모리 대역폭을 (PIM의 경우 각 Vault의 대역폭의 평균 값, Host CPU의 경우 메인 메모리의 대역폭) 나타낸다. 여기서 수행시간 동안 평균적인 메모리 대역폭 활용도의 경우 PIM 기반 시스템이 싱글 코어 및 멀티코어 Host 시스템 대비 약 5.58배 및 약 1.21배 높은 활용률을 보여준다.

각 Host (비순차 코어 1개) CPU + PIM (PIM Vault 4개)으로 구성된 시스템과 순차형 싱글 코어 및 5개 (1+4)개의 순차형 멀티코어로 구성된 시스템 상에서의 실행 시간 및 성

능 비교를 나타내었으며, Host + PIM 항목의 경우 모든 vault에서의 대역폭을 누적한 것이 아닌, 각 vault 별 평균 대역폭을 나타낸 것이다. 따라서, 실질적인 PIM Device 상에서의 메모리 대역폭의 경우 해당 수치의 약 4배(PIM vault 개수 고려)임을 감안할 필요가 있다.

Fig. 6는 Fig. 5에서 상대적으로 표현된 수행 시간을 Host + PIM으로 구성된 시스템의 수행 시간을 기준으로 정규화 한 결과이다. 여기에서는 순차실행 프로세서로 구성된 Host 시스템과, 같은 조건으로 순차실행 프로세서 기반의 연산 유닛에 기반한 PIM과의 수행성능을 비교하였다. 기존의 전통적인 메모리 계층 구조 상의 시스템에서는 다중 메모리 컨트롤러를 CPU에 도입하지 않을 경우, 여러 개의 코어가 하나의 메인 메모리 및 메모리 컨트롤러 자원을 공유하는 형태로 구동된다.

그리고 순차실행 프로세서의 경우, lock-free 캐시구조가 아닌, blocking cache의 적용으로 인해, 캐시 미스 발생 이후에 발생하는 연속적인 캐시 접근 요청에 대해서 대응할 수 없기 때문에, 메모리 수준의 병렬성을 활용하기 힘들다는 단점이 존재한다. 따라서, Fig. 6와 같이 STREAM 벤치마크에서 발생하는 working memory space에 대한 thrashing 현상이 빈번한 워크로드들을 수행함에 있어, 여러 계층의 blocking 캐시로 구성된 시스템의 경우, 같은 연산 유닛의 숫자에 기반한 PIM 시스템보다 더 좋지 않은 성능을 보이는 경우가 발생한다. Host+PIM(4 vaults) 환경의 경우 싱글 코어 대비 약 9.5배, 그리고 같은 종류의 순차코어 5개로 구성된 시스템 대비 약 4.54배 더 높은 성능을 보인다.

4. 결 론

본 연구는 최근 대두되고 있는 데이터 집약적 프로세싱을 위한 시스템 중 하나인, 프로세싱 인 메모리 아키텍처의 성능적 평가를 위한 소프트웨어 시뮬레이션 환경인 TP-Sim의 설계와 구현을 수행하였다. 본 논문에서 제안한 시뮬레이터의 경우 성능 평가에서 주로 다루어지는 실행 시간, 메모리 대역폭 등의 수행 결과 등을 산출할 수 있으며, 추가적으로 Host CPU 및 PIM vault 내부의 캐시와 관련된 통계 값을 통하여 PIM 및 워크로드 특성 간의 연관된 상관 관계 등을 탐색할 수 있는 기능들도 제공한다. 이를 통해, 추후 새로운 형태의 PIM 구조 및 관리 기법 등에 대한 성능 평가 및 검증 등을 진행할 수 있다.

또한 본 연구에서는 메모리 대역폭 측정을 위하여 간단한 벡터 연산 커널들을 포함하고 있는 STREAM 벤치마크의 구동을 통해서, 시뮬레이터를 통한 여러 형태의 시스템들에 대한 성능을 확인해보고자 하였다. 전통적인 메모리 구조에 기반한 시스템 및 PIM 모델과의 비교를 통해, 해당 벤치마크의 구동에 있어서 싱글 코어 및 5코어 멀티 코어 대비 약 9.5배 및 4.54배 더 높은 수행 성능을 보였으며, 대역폭 활용 측면에서도 일반적인 멀티코어 및 싱글 코어 시스템 대비 약 5.58배 및 1.21배라는 높은 활용률을 보여주었다.

차후 연구에서는 본 연구에서 진행한 TP-Sim을 확장하여 여러 개의 PIM 장치들을 연결할 경우 발생하는 inter-connection network에 대한 시뮬레이션 기능에 대한 추가와 더불어, 범용적 연산 유닛 (general purpose processing unit)이외에도 일부 연산(간단한 산술 연산 및 MAD, MAC)만을 제공하는 형태의 fixed function processing unit에 기반한 PIM 장치에 대한 모델링도 가능하게 하고자 한다.

그리고 최종적으로는 최근의 데이터 집약적 프로그램들을 위해 구성되는, 멀티코어 CPU, GPGPU, 그리고 PIM 등으로 구성되는 이기종 프로세싱 시스템에서의 성능 검증이 가능한 시뮬레이션 프레임워크 구현의 토대를 마련하고자 한다.

감사의 글

This research was supported by National Research Foundation of Korea (NRF) Grant funded by the Korean Government (Ministry of Education) NRF-2021R1I1A1A01059737.

참고문헌

1. Mason, Tony, et al. “Unexpected performance of Intel® Optane™ DC persistent memory.” IEEE Computer Architecture Letters 19.1 (2020): 55-58.
2. Wuu, John, et al. “3D V-Cache: the Implementation of a Hybrid-Bonded 64MB Stacked Cache for a 7nm x86-64 CPU.” 2022 IEEE International Solid-State Circuits Conference (ISSCC). Vol. 65. IEEE, 2022.
3. Boroumand, Amirali, et al. “Google workloads for consumer devices: Mitigating data movement bottlenecks.” Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems. 2018.
4. Ke, Liu, et al. “Near-memory processing in action: Accelerating personalized recommendation with axdimm.” IEEE Micro 42.1 (2021): 116-127.
5. He, Mingxuan, et al. “Newton: A DRAM-maker’s accelerator-in-memory (AiM) architecture for machine learning.” 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2020.
6. Pawlowski, J. Thomas. “Hybrid memory cube (HMC).”

2011 IEEE Hot chips 23 symposium (HCS). IEEE, 2011.
7. Gómez-Luna, Juan, et al. “Benchmarking memory-centric computing systems: Analysis of real processing-in-memory hardware.” 2021 12th International Green and Sustainable Computing Conference (IGSC). IEEE, 2021.
8. Yu, Chao, Sihang Liu, and Samira Khan. “Multipim: A detailed and configurable multi-stack processing-in-memory simulator.” IEEE Computer Architecture Letters 20.1 (2021): 54-57.
9. https://github.com/CMU-SAFARI/ramulator-pim
10. Sanchez, Daniel, and Christos Kozyrakis. “ZSim: Fast and accurate microarchitectural simulation of thousand-core systems.” ACM SIGARCH Computer architecture news 41.3 (2013): 475-486.
11. Luk, Chi-Keung, et al. “Pin: building customized program analysis tools with dynamic instrumentation.” Acm sigplan notices 40.6 (2005): 190-200.
12. McCalpin, John D., 1995: “Memory Bandwidth and Machine Balance in Current High Performance Computers", IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, December 1995.
13. Gao, Mingyu, Grant Ayers, and Christos Kozyrakis. “Practical near-data processing for in-memory analytics frameworks.” 2015 International Conference on Parallel Architecture and Compilation (PACT). IEEE, 2015.
14. Min, Chuhan, et al. “NeuralHMC: An efficient HMC-based accelerator for deep neural networks.” Proceedings of the 24th Asia and South Pacific Design Automation Conference. 2019.

접수일: 2023년 8월 18일, 심사일: 2023년 9월 15일,
게재확정일: 2023년 9월 19일